

DATA PROCESSOR CONTROLLED INTERFACE WITH
MULTIPLE TREE OF ELEMENTS VIEWS EXPANDABLE
INTO INDIVIDUAL DETAIL VIEWS

Cross-Reference to Related Copending Patent Applications

5 The following patent applications which are assigned
to the assignee of the present invention cover subject
matter related to the subject matter of the present
invention: "Data Processor Controlled Display System
With a Plurality of Selectable Basic Function Interfaces
10 for the Control of Varying Types of Customer Requirements
and With Additional Customized Functions", Attorney
Docket No. AM9-97-153; "Data Processor Controlled Display
System With a Plurality of Switchable Customized Basic
Function Interfaces for the Control of Varying Types of
15 Operations", Attorney Docket No. AM9-97-155; "Data
Processor Controlled Display System for the Control of
Operations With Control Properties Which are Selectably
Constant or Variable", Attorney Docket No. AM9-97-156;
"Data Processor Controlled Display Interface With Tree
20 Hierarchy of Elements View Expandable into Multiple
Detailed Views", Attorney Docket No. AM9-97-157; "Data
Processor Controlled Display With a Tree of Items With
Modification of Child Item Properties Designated at
Parent Level Without Modification of Parent Item
25 Properties", Attorney Docket No. AM9-97-159; and "Data
Processor Controlled Display System With a Tree Hierarchy
of Elements View Having Virtual Nodes", Attorney Docket
No. AM9-97-160; all are assigned to International
Business Machines Corporation by Claudia Alimpich et al.
30 and all are filed concurrently herewith.

44250 64059860

Technical Field

The present invention relates to user-friendly interactive computer supported display technology and particularly to such user interactive systems and
5 programs which have user interactive interfaces with tree hierarchies of objects.

Background of the Invention

The computer and computer related industries have benefitted from a rapidly increasing availability of data
10 processing functions. Along with this benefit comes the problem of how to present the great number and variety of available functions to the interactive operator or user in display interfaces which are relatively easy to use. In recent years, the hierarchal tree has been a widely
15 used expedient for helping the user to keep track of and organize the operative and available functions. In typical tree structures such as those in Microsoft Windows 95^(TM) and IBM Lotus^(TM) systems, there is presented on the display screen a variety of operating and
20 available functions and resources in tree hierarchies with classes and subclasses of functions and resources displayed as objects or elements at nodes in a descending and widening order based upon some kind of derivation from the next higher class or subclass. In conventional
25 tree displays, it is customary to show the tree on one region of the screen and to permit the user to interactively select any object on the screen for a detail view showing attributes or properties of the selected object or of child objects of the selected
30 object. Such child objects may be said to be derived or comprehended by the selected or parent object. This detail view is customarily displayed in a second region

7044250 24053850

of the screen which is separate from the tree view. These conventional interfaces operate effectively with relatively simple tree structures, particularly when the tree is small enough to be displayed within the portion
5 of the screen allocated for the tree view. However, with the great increase of functions available in computers, it is now often the situation that the displayed trees will be greatly expanded and have many nodes and branches. With these more complex trees, the entire tree
10 may not be viewed in the tree view region of a screen and extensive scrolling has to be done in order to view all portions of the tree. This complicates user interaction, particularly in situations where the user needs to view and compare objects or properties of objects which are in
15 portions of the tree remote from each other. Also, when the user needs to move objects from one part of a tree to another part of the tree, the inability to view source and destination points of the potential object transfer simultaneously on the screen complicates the user
20 interaction.

It should also be noted that the tree hierarchical structure is one of the primary organizational expedients in the object oriented programming processes which have become more prevalent over the past decade. In object
25 oriented programming systems there are many implementations in which there are hierarchies of child objects in subclasses of parent object classes wherein the child objects inherit at least some attributes of the parent objects. The present invention may be
30 advantageously used with such tree hierarchies. Like the trees described above, it is not unusual in object oriented programming to have very elaborate and extensive hierarchal relationships displayed in tree views.

Before proceeding with the description of the invention, it is important that we clarify the terminology used in describing the present invention with respect to terminology used in the art. The typical tree structure used in interactive displays has a plurality of levels of nodes at which elements are displayed. Such elements may be in text or icon form. The elements each represent a function or resource of the system or program which is to be interactively accessed through the displayed tree. It is customary to refer to these displayed elements as objects so as to conveniently use the terms child objects or parent objects in describing various hierarchical relationships between objects. It should be understood that in object oriented programming the term object is, of course, extensively used to represent the basic program units. Thus, it may sometimes be the case that the object in a displayed tree does represent an object in an object oriented program. However, since trees of displayed objects are used to interface with many systems not involving object oriented programming, the use of the word object is not intended to limit the description or claim to object oriented applications. The terms object and elements as used in the claims should be considered as substantially equivalent.

Summary of the Invention

The present invention provides a solution to the above problems by providing the user with the ability to simultaneously show two different tree views. The invention relates to a data processor controlled display system providing an ease of use interface based upon a tree view of selectable elements arranged in a hierarchy

on a display screen. It involves a first tree view of a hierarchy of selectable elements or objects in one region of a display interface screen and a second tree view of a hierarchy of selectable elements or objects in a second region of a display interface screen in combination with means responsive to the selection of an object for displaying additional data relative to said selected object in another region of said screen. This additional data is provided in what may be referred to as a detail view of the selected object. This detail view may set forth attributes of the selected object or of the child objects of the selected object. In this manner, the user may get whatever details he requires with respect to objects in either tree view, after which he may redistribute or allocate his resources by transferring objects within each tree view or between tree views. Such transfer may be done by drag and drop means. The two different tree views may be of respective portions of the same hierarchal tree or the views may be of different trees.

As will be seen hereinafter in greater detail, the present invention may be advantageously used in the control of production operations such as high throughput printing operations.

Brief Description of the Drawings

Fig. 1 is a block diagram of an interactive data processor controlled display system including a central processing unit which is capable of implementing the program of the present invention for presenting object tree views and the detail views;

Fig. 2 is a diagrammatic view of a display screen on which two separate hierarchical object tree views are shown prior to any user selection;

Fig. 3 is the view of the display screen of Fig. 2 showing the details view resulting from the selection of an object from the first tree view;

Fig. 4 is the view of the display screen of Fig. 2 showing the details view resulting from the selection of an object from the second tree view;

Fig. 5 is the display screen of Fig. 4 after the selection of an object in the second tree view to be transferred and thus allocated to an object in the first tree view;

Fig. 6 is the display screen of Fig. 5 showing the movement of the object selected for transfer at an intermediate point in a drag and drop transfer;

Fig. 7 is the display screen of Fig. 6 upon the completion of the drag and drop transfer;

Fig. 8 is a flowchart showing the development of the program of the present invention for displaying two or more tree views and the detail view of the attributes of selected objects from said tree views; and

Fig. 9 is a flowchart showing the running of the program described with respect to Fig. 8.

25 Detailed Description of the Preferred Embodiment

Referring to Fig. 1, a typical data processing system is shown which may function as the computer controlled display terminal used in implementing the tree view and detail view functions of the present invention.

30 A central processing unit (CPU), such as one of the PC microprocessors available from International Business Machines Corporation, is provided and interconnected to

various other components by system bus 12. An operating system 41 runs on CPU 10, provides control and is used to coordinate the function of the various components of Fig. 1. Operating system 41 may be one of the commercially available operating systems such as DOS or the OS/2 operating system available from International Business Machines Corporation (OS/2 is a trademark of International Business Machines Corporation); Microsoft Windows 95^(TM) or Windows NT^(TM), as well as Unix and AIX operating systems. A programming application for displaying multiple hierarchical tree views and for presenting detail views in accordance with the present invention, application 40 to be subsequently described in detail, runs in conjunction with operating system 41 and provides output calls to the operating system 41 which implements the various functions to be performed by the application 40.

A read only memory (ROM) 16 is connected to CPU 10 via bus 12 and includes the basic input/output system (BIOS) that controls the basic computer functions. Random access memory (RAM) 14, I/O adapter 18 and communications adapter 34 are also interconnected to system bus 12. It should be noted that software components, including the operating system 41 and the application 40, are loaded into RAM 14 which is the computer system's main memory. I/O adapter 18 may be a small computer system adapter that communicates with the disk storage device 20, i.e. a hard drive. Communications adapter 34 interconnects bus 12 with an outside network enabling the data processing system to communicate with other such systems of which objects may also be displayed in the object tree interfaces of the present invention. I/O devices are also connected to

system bus 12 via user interface adapter 22 and display adapter 36. Keyboard 24, trackball 32, mouse 26 and speaker 28 are all interconnected to bus 12 through user interface adapter 22. It is through such input devices

5 that the user interactive functions involved in the displays of the present invention may be implemented. Display adapter 36 includes a frame buffer 39 which is a storage device that holds a representation of each pixel on the display screen 38. Images may be stored in frame

10 buffer 39 for display on monitor 38 through various components such as a digital to analog converter (not shown) and the like. By using the aforementioned I/O devices, a user is capable of inputting information to the system through the keyboard 24, trackball 32 or mouse

15 26 and receive output information from the system via speaker 28 and display 38. In the illustrative embodiment, which will be subsequently described, the multiple hierarchical trees of objects and the subsequent separated details views of different objects will be

20 shown with respect to the control of high throughput printers such as electrophotographic or laser printers. A local printer system 44 may be accessed and controlled via printer adapter 43 while, as previously mentioned, networked printers may communicate via communications

25 adapter 34.

There will now be described a simple illustration of the present invention with respect to the display screens of Figs. 2 through 7. When the screen images are described, it will be understood that these may be

30 rendered by storing an image and text creation programs, such as those in any conventional window operating system in the RAM 14 of the system of Fig. 1. The operating system is diagrammatically shown in Fig. 1 as operating

system 41. The display screens of Figs. 2 through 7 are presented to the viewer on display monitor 38 of Fig. 1. In accordance with conventional techniques, the user may control the screen interactively through a conventional I/O device such as mouse 26 of Fig. 1 which operates through user interface 22 to call upon programs in RAM 14 cooperating with the operating system 41 to create the images in frame buffer 39 of display adapter 36 to control the display on monitor 38.

The display screen of Fig. 2 shows a first portion of a hierarchical tree 52 in region 50 which represents a hierarchy of available printers involved in the printer production operations being controlled. Also shown is a second tree portion 53 in screen region 51 which is a list of retained jobs awaiting assignment to printers or other disposal. It should be noted that the objects in tree portions 52 and 53 may be parts of the same overall tree. With such an overall tree where the user wishes to do a side by side comparison of a plurality of resources, needs and functions associated with nodes in different parts of the tree, he would scroll to his desired first portion of the tree in region 50 and then scroll to his desired second portion 53 in the overall tree in the other window or region 51. While we are showing only two regions, 50 and 51, for convenience of illustration, it is understood that any reasonable plurality could be set up to handle a corresponding plurality of tree portions. Also, regions 50 and 51 could be expanded or curtailed in size dependent upon the size of the selected tree portions. This changing of the sizes of regions 50 and 51 could be done by using any of the conventional window size changing techniques provided by any of the above-mentioned operating systems. Also, under appropriate

F040250 "E4059360

conditions, tree portions 52 and 53 could be from different trees. In the present example the comparison of tree portions 52 and 53 is being carried out so that the user may decide which of the retained jobs in tree 53 may be assigned to which of the printers in tree 52. In order for the user controlling the printing operation to assign jobs to printers, he may need further information: expanded views giving details of the jobs in tree portion 53 and the printers in tree portion 52. Region 54 in the screen of Fig. 2 is provided for such expanded views. Assume that he first wishes available printer details, as shown in Fig. 3, he selects Printer object 59 which becomes highlighted, and he gets the expanded or detail view 56 of all of Printer1 through Printer4 which may be considered the child objects of Printer object 59. The values of the attributes 55 of the Printer's child objects are set forth. Similarly, if the user now wishes to get an expanded or detail view of the Retained Jobs, he clicks on Retained Jobs object 69, Fig. 4, which becomes highlighted and he gets the expanded or detail view 58 of all of the Jobs8 through Jobs10 which may be considered the child objects of Retained Jobs object 69. The values of the attributes 57 of the job child objects are set forth. Now, after the user has seen the detail views of his resources, he may commence to assign the Jobs8 through Jobs10 to Printer1 through Printer4. He may do this by a "Drag and Drop" screen interaction. In Fig. 5, for example, he points to Job10 with pointer 60, clicks on it and it becomes movable. Then, Fig. 6, through the movement of pointer 60, he drags Job10 along path 62 to deposit Job10 on Printer1 and thus indicate that Job10 is assigned to Printer1. As a result, Fig. 7, the allocation of Job10 has been disposed of and it is no

longer in tree view 53 or in details view 58 of jobs. The remaining jobs may be similarly allocated to printers.

Now with reference to Fig. 8, we will describe a process implemented by a program according to the present invention. The program is continuous and involves the development of the display screen interfaces previously described with respect to Figs. 2 through 7. In the flowchart of Fig. 8, a basic hierarchical interface is set up, step 80, wherein the objects represent the functions and resources of the systems. In the present example, these would be printer operation control interfaces. Of course, appropriate conventional linkages are set up between representations of functions displayed on a screen whether these representations be text or icons representative of the functions and the functions themselves, step 81. Then, steps 82 and 83, first and second regions are set up, e.g. regions 50 and 51, Fig. 2, within which selected portions of trees may be viewed. In step 84 of Fig. 8, scrolling means are provided to scroll the tree hierarchies within regions 50 and 51 to set the desired tree portions in the two regions e.g., tree portions 52 and 53, Fig. 2. Next, step 85, Fig. 8, a process is set up responsive to the selection of a particular object for detail views for the object. This would, for example, be the detail views 56 of printers, Fig. 3, and the detail views 58 of jobs, Fig. 4. Then, step 86, a process is set up for the selection of an object for transfer e.g., Job10, Fig. 5, for moving the selected object, step 87, out of its tree view and into the other tree view as illustrated in Fig. 6, and then, step 88, upon completion of the transfer, deleting the object from its original tree position as illustrated in

FOI 250 E 4053860

Fig. 7. It should be noted in connection with setting up the programming for carrying out steps 86 through 88 that most of the systems mentioned above, such as OS/2 or Windows 95 have drag and drop functions which are readily modifiable to carry out the described transfer between tree portions.

Now that the basic program has been described and illustrated, there will be described with respect to Fig. 9, a flow of a simple operation showing how the program could be run. First, steps 90 and 91, the display of Fig. 2 is set up with trees 52 and 53 respectively in regions 50 and 51. Next, an initial determination is made as to whether an object in the tree has been selected for an expanded view which, in the present example, would be detail view of its child objects, step 92. If No, the flow is routed to decision step 94. If there is a selection of a particular object in either trees 52 or 53, then, step 93, a detail view of the attributes of the child objects of the selected object is shown as in Figs. 3 and 4. After viewing the detail views, the user would make decisions with respect to transferring objects from one tree view to another. A determination is made, decision step 94, as to whether an object has been selected for transfer. If Yes, then step 96, the object is picked by the pointer, Fig. 5, dragged from the original tree view, step 97, Fig. 6, to the target object in the other tree, step 98. The child object, which was transferred, is then dropped from its original tree, step 99, as in Fig. 7, and the flow is returned to decision step 92 via branch "A". If the decision from step 94 is No child object has been selected for transfer, then the flow is branched back to

step 92 to await the user selection of other tree views in accordance with the program of this invention.

While the embodiment described has been for a hierarchical tree used in the control of resources in a printing environment, it has been mentioned hereinabove that the present invention may be advantageously used in object oriented programming systems where hierarchical trees are extensively used. By way of background, with respect to object oriented programming, its techniques involve the definition, creation, use and construction of "objects". These objects, as mentioned above, are not to be confused with display tree objects or elements; they are software entities comprising data elements or attributes and methods, which manipulate the data elements. The data and related methods are treated by the software as an entity and can be created, used and deleted as such. The data and functions enable objects to model their real world equivalent entity in terms of its attributes, which can be presented by the data elements, and its behavior which can be represented by its methods. Objects are defined by creating "classes" which are not objects themselves, but which act as templates which instruct a compiler how to construct the actual object. For example, a class may specify the number and type of data variables and the steps involved in the functions which manipulate the data. An object is actually created in the program by means of a special function called a constructor which uses the corresponding class definition and additional information, such as arguments provided during object creation, to construct the object. A significant property of object oriented programming is inheritance, which allows program developers to reuse pre-existing

programs. Inheritance allows a software developer to define classes and the objects which are later created from them as related through a class hierarchy.

Specifically, classes may be designated as subclasses of other base classes. A subclass inherits and has access to all of the public functions of its base classes as though these functions appeared in the subclass.

Alternatively, a subclass can override some or all of its inherited functions or may modify some or all of its inherited functions by defining a new function with the same form. The creation of a new subclass borrowing the functionality of another class allows software developers to easily customize existing code to meet their particular needs.

In order to help object oriented program developers construct programs, particularly when inheritance is involved, object oriented programming systems make use of hierarchical trees in order to help the interactive programmers and users in the understanding of the basic structures of classes and subclasses, as well as parent objects in classes and their child objects in the various types of subclasses inheriting some of their properties from the classes above them in the hierarchy. Object oriented technology trees are at times very elaborate with many branches and trunks in their hierarchy. It is in the organization and reorganization of object oriented program trees that the present invention may be effectively used when it would be desirable to study the relationships of various portions of such programs to each other or to the tree hierarchies of other related programs.

Although certain preferred embodiments have been shown and described, it will be understood that many changes and modifications may be made therein without departing from the scope and intent of the appended
5 claims.

704250 "E4093650